משרד החינוך

תלת"ל -

תקינה לשיתוף תוכני למידה

מבוססת SCORM

תקן למערכות LMS

גרסה: טיוטה 1.1-7

תאריך:1-06-2014

מחבר: משרד החינוך

דוא"ל לשאלות: tkina@education.gov.il

אתר מרכז: אתר משרד החינוך

היסטוריית גרסאות

מספר הגרסה	תאריך	שם הפרק /ים המעודכנים	שונה על ידי	שינויים ודגשים עיקריים
1.1-7 טיוטה	1 ביוני 2014	תקן למערכות LMS	משרד החינוך	התווסף נספח כ' "SCORM API"

מסמך זה מהווה טיוטה בלבד ומפורסם למטרת קבלת הערות מקצועיות ולא לכל מטרה אחרת.

תקן למערכות LMS
של משרד החינוך

טיוטה 1.1-7

נספח כ': SCORM API

(לתכנים המופעלים בתצורת הפעלה אופליין)

תוכן עניינים

1. מבוא 4

2. Session Methods 4

2.1 **Initialize** 4

2.2 **Terminate** 4

3. Data-Transfer Methods 5

3.1 **GetValue** 5

3.2 **SetValue** 5

3.3 **Commit** 5

4. Support Methods 6

4.1 GetLastError 6

4.2 GetErrorString 6

4.3 GetDiagnostic 6

5. מודל הפעלה 6

6. API INSTANCE 8

6.1 **חשיפת המימשק** 8

6.2 **גרסת אובייקט ה-API INSTANCE** 9

6.3 **קוד לדוגמה למציאת אובייקט ה-API INSTANCE מתוך ה-SCO** 9

6.4 **דוגמה נוספת** 10

פירוט הנושאים

1. מבוא

בנספח זה מוגדר ממשק של פונקציות שבהן ה-SCO יוכל להשתמש. את הפונקציות האלו ה-LP יממש ויחשוף אותן באמצעות API INSTANCE אל ה-SCO.

הפונקציות בממשק מתחלקות ל-3 קבוצות:

א. Session Methods – פונקציות אלו נועדו להגדיר תחילת וסוף התקשרות בין ה- SCO ל-LMS.
ב. Data-transfer Methods – פונקציות אלו נועדו בכדי לעדכן את ערכי משתני ה-Data Model.
ג. Support Methods – פונקציות אלו נועדו לטיפול בשגיאות.

יש לשים לב כי -

- שמות הפונקציות רגישים לאותיות קטנות/גדולות (case sensitive)
- כל הפרמטרים של הפונקציות הם case sensitive
- כל הפרמטרים מועברים כמחרוזת, ועליהם לעמוד בהגדרת הטיפוסים של ה-DM, כפי שמוגדר בנספח OFFLINE DM.
- התקשורת בין ה-SCO ל-LMS הינה חד צדדית. כלומר, ה-SCO הוא היחיד שיוזם פניות ל-LMS ולא להיפך. ה-LMS לעולם לא יוזם קריאה ל-SCO.

2. Session Methods

2.1 Initialize

פונקציה זו מגדירה את תחילת ההתקשרות בין ה- SCO ל-LMS.
חתימה: boolean Initialize()
פרמטרים: ללא.
ערך החזרה: true/false:
true – אם הפעולה הצליחה ע"י ה-LMS.
false – אם הפעולה נכשלה ע"י ה-LMS. כמו כן יוגדר קוד שגיאה מתאים.

2.2 Terminate

פונקציה זו מגדירה את סיום ההתקשרות בין ה- SCO ל-LMS.
כמו כן, זהו בעצם גם קריאה מרומזת ל-COMMIT – שכן ה-LMS מחוייב לשמור את כל המידע מאז הקריאה האחרונה לCOMMIT / תחילת ההפעלה.
חתימה: Boolean Terminate()
פרמטרים: ללא.
ערך החזרה: true/false:
true – אם הפעולה הצליחה ע"י ה-LMS.
false – אם הפעולה נכשלה ע"י ה-LMS. כמו כן יוגדר קוד שגיאה מתאים.

3. Data-Transfer Methods

3.1 GetValue

פונקציה זו נועדה לבקש מידע מה-LMS.
חתימה: return_value GetValue(parameter)
פרמטרים: מחרוזת המזהה את שם האלמנט מה-DM שאותה מבקש לקבל ה-SCO.
ערך החזרה:
אם אין שגיאה – יוחזר ערך האלמנט שאותו ביקש ה-SCO.
אם יש שגיאה – תוחזר מחרוזת ריקה "" ויוגדר קוד שגיאה מתאים.
יש לשים לב שייתכן ותוחזר מחרוזת ריקה ולא הייתה שגיאה (למשל זהו ערכו של משתנה כלשהו)

3.2 SetValue

פונקציה זו נועדה לעדכן מידע מזמן ריצת ה-SCO ב-LMS.
המידע ישמר מיד בצד השרת ב- LMS או ימתין ב-cache בצד הלקוח עד לפעולת Commit/Terminate.
חתימה: return_value SetValue(parameter1, parameter2)
פרמטרים:
parameter1 – מחרוזת המזהה את שם האלמנט מה-DM שאותה מבקש לעדכן ה-SCO.
parameter2 – הערך שאותו מבקש ה- SCO לשים באלמנט ה-DM לפי הזיהוי של parameter1.
ערך החזרה: true/false:
true – אם הפעולה הצליחה ע"י ה-LMS. (כלומר הושם ערך parameter2 באלמנט ה-dm המוזהה ע"י השם שהוגדר ב-parameter1)
false – אם הפעולה נכשלה ע"י ה-LMS. כמו כן יוגדר קוד שגיאה מתאים.

3.3 Commit

פונקציה זו נועדה בכדי להבטיח שכל המידע מאז הקריאה האחרונה ל-Initialize או ל-Commit, נשמר בצד השרת. ולכן – אם קיים cache בצד לקוח, בעת קריאה לפונקציה זו יועבר כל ה-cache אל ה-LMS ויישמר בו. אם לא קיים cache אז הפונקציה לא תעשה דבר.
חתימה: return_value Commit()
פרמטרים:
ללא
ערך החזרה: true/false:
true – אם הפעולה הצליחה ע"י ה-LMS. כלומר – אם קיים cache אזי הוא עובר ונשמר, אחרת תמיד יוחזר ערך זה.
false – אם הפעולה נכשלה ע"י ה-LMS. כלומר הcache לא הצליח לעבור ל-LMS. כמו כן יוגדר קוד שגיאה מתאים.

4. Support Methods

4.1 GetLastError

פונקציה זו מחזירה את קוד השגיאה האחרונה שקרתה במהלך התקשורת של ה- SCO עם ה-LMS. השימוש הנכון בפונקציה זו היא לבדוק האם פונקציה מSession/Data-transfer Methods לא הצליחה (כלומר החזירה false) ואז לבקש את קוד השגיאה בעזרת פונקציה זו.

חתימה: int GetLastError()

פרמטרים: ללא

ערך החזרה: קוד השגיאה כמחרוזת הניתן להמרה למספר שלם בתחום 0 עד 65536 כולל.

4.2 GetErrorString

פונקציה זו מחזירה את תיאור השגיאה האחרונה שקרתה במהלך התקשורת של ה- SCOעם ה-LMS.

חתימה: return_value GetErrorString(parameter)

פרמטרים: קוד השגיאה (integer) שמבקשים לקבל את תיאורה.

ערך החזרה: תיאור השגיאה (מחרוזת) המתאים לקוד השגיאה שהתקבל כפרמטר.

ערך ההחזרה לא יעלה על 255 תווים.

אם ה-LMS אינו מכיר את קוד השגיאה שהתקבל כפרמטר, יוחזר הערך "" (מחרוזת ריקה).

4.3 GetDiagnostic

פונקציה זו נועדה לשימוש ספציפי של ה-LMS. פונקציה זו מאפשרת להעביר מידע דיאגנוסטי נוסף באמצעות ה-API INSTANCE.

חתימה: return_value GetDiagnostic(parameter)

פרמטרים: מחרוזת שלא עולה על 255 תווים. ערך תלוי מימוש – לדוגמה קוד שגיאה.

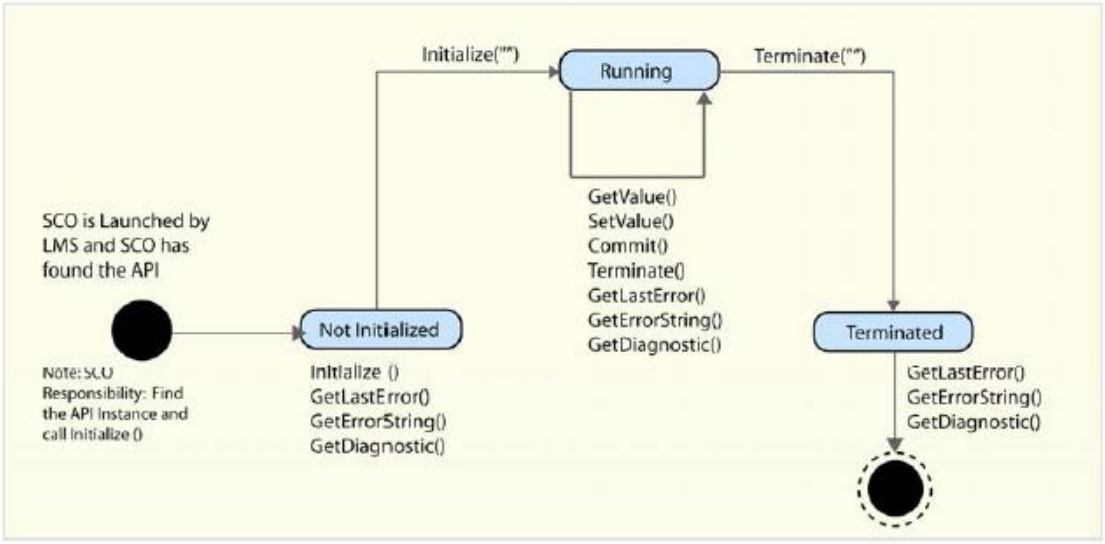
ערך החזרה: מידע דיאגנוסטי (מחרוזת).

ערך ההחזרה לא יעלה על 255 תווים.

אם ה-LMS אינו מכיר את ערך הפרמטר שהתקבל, יוחזר הערך "" (מחרוזת ריקה).

אם הפרמטר שהתקבל הוא "" (מחרוזת ריקה) מומלץ להחזיר מידע אודות השגיאה האחרונה.

5. מודל הפעלה



לצורך המחשת השימוש הנכון בפונקציות ה-API, נחלק את תהליך ההתקשורת של ה- SCO עם ה-LMS לשלושה מצבים קונספטואלים:

1. Not initialized – מצב זה מוגדר בין הצגת ה-SCO לבין הקריאה לפונקציה Initialize מתוך ה-SCO.

2. Running – מצב זה מוגדר בין קריאת ה-SCO לפונקציה Initialize דרך הפעלתו ועד לקריאה לפונקציה Terminate עם סיום ההפעלה.

3. Terminated – מצב זה מוגדר לאחר הקריאה לפונקציה Terminate ע"י ה-SCO.

בכל אחד מהמצבים הנ"ל, ה-SCO יכול לקרוא לפונקציה כמתואר בתרשים.

6. API INSTANCE

6.1 חשיפת המימשק

כחלק מתקן תלת"ל, ה-LMS מחוייב לחשוף את ה-SCORM API כאובייקט DOM נגיש ל-SCO בשם API_1484_11.
ממשק זה צריך להיות נגיש ל-SCO באמצעות שפת ECMAScript (לרוב JS).
ה-SCO צריך להיות מופעל במיקום מסויים בהיררכית ה-DOM, כך שה-SCO יהיה בחלון/פריים בן של חלון שמכיל את ה-API INSTANCE.

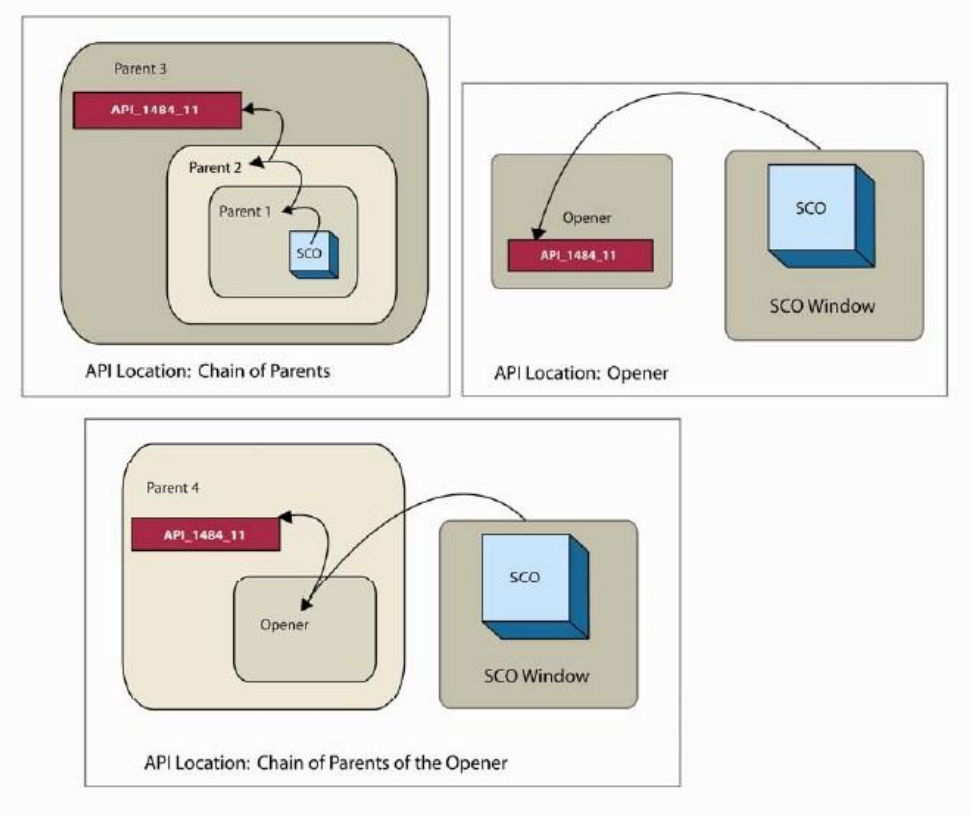


Figure 3.2.1a: Permitted DOM Locations of an API Implementation

ישנן 3 אפשרויות למיקום ה-SCO ביחס ל-API INSTANCE:

1. "Chain of Parents" – באפשרות זו, ה- SCO יופעל במבנה היררכי של פריימים. אובייקט ה-API INSTANCE ימוקם בפריים שנמצא בהיררכיה של אחד מפריימי האב של הפריים שבו מופעל ה-SCO.
2. "Opener" – באפשרות זו, ה-SCO מופעל בחלון חדש ("פופאפ") כך שבחלון שפתח אותו נמצא אובייקט ה-API INSTANCE.
3. "Chain of Parents of the Opener" – באפשרות זו, ה- SCO מופעל בחלון חדש, כך שאובייקט ה-API INSTANCE ימוקם בפריים שנמצא בהיררכיה של אחד מפריימי האב של החלון שפתח את חלון ה-SCO.

6.2 גרסת אובייקט ה-API INSTANCE

כחלק מתקן תלת"ל, לאובייקט ה-API INSTANCE (API_1484_11) צריך להיות מאפיין בשם "version" המגדיר את גרסת ה-API INSTANCE. ערך מאפיין זה יוגדר להיות המחרוזת "1.0".

6.3 קוד לדוגמה למציאת אובייקט ה-API INSTANCE מתוך ה-SCO

```
var nFindAPITries = 0;
var API = null;
var maxTries = 500;
var APIVersion = "";

// The ScanForAPI() function searches for an object named API_1484_11
// in the window that is passed into the function. If the object is
// found a reference to the object is returned to the calling function.
// If the instance is found the SCO now has a handle to the LMS
// provided API Instance. The function searches a maximum number
// of parents of the current window. If no object is found the
// function returns a null reference. This function also reassigns a
// value to the win parameter passed in, based on the number of
// parents. At the end of the function call, the win variable will be
// set to the upper most parent in the chain of parents.
function ScanForAPI(win)
{
   while ((win.API_1484_11 == null) && (win.parent != null) && (win.parent != win)) {
      nFindAPITries++;
      if (nFindAPITries > maxTries) {
         return null;
      }
      win = win.parent;
```

```
        }
        return win.API_1484_11;
    }

    // The GetAPI() function begins the process of searching for the LMS
    // provided API Instance. The function takes in a parameter that
    // represents the current window. The function is built to search in a
    // specific order and stop when the LMS provided API Instance is found.
    // The function begins by searching the current window's parent, if the
    // current window has a parent. If the API Instance is not found, the
    // function then checks to see if there are any opener windows. If
    // the window has an opener, the function begins to look for the
    // API Instance in the opener window.
    function GetAPI(win) {
        if ((win.parent != null) && (win.parent != win)) {
            API = ScanForAPI(win.parent);
        }
        if ((API == null) && (win.opener != null)) {
            API = ScanForAPI(win.opener);
        }
        if (API != null) {
            APIVersion = API.version;
        }
}
```

6.4 דוגמה נוספת

דוגמה נוספת פורסמה באתר המשרד בנספח ט"ז בקובץ
Scos/Scripts/APIWrapper.js (הפונקציה getAPI() בשורה 482)

שימו לב כי - רשימת הנספחים מתפרסמת באתר המשרד ומתעדכנת מעת לעת